Dynamic Programming

Series of decisions

$\Downarrow$

Recursive problem $\leftarrow$ English

$\Downarrow$

Recurrence / recursive algo $\leftarrow$ Math

$\Downarrow$

Iterative evaluation

Is $A[j]$ an element of the longest increasing subsequence?

We want $LIS(0,1)$

$LIS(i,j) :=$ length of the longest increasing subsequence of $A[j\,..\,n]$ with all elements larger than $A[i]$

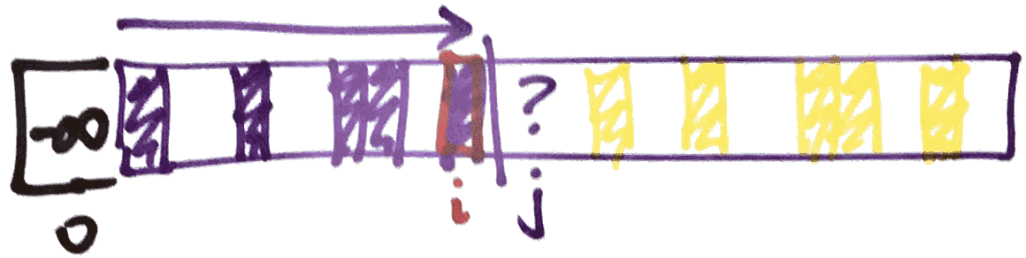
$$LIS(i,j) = \begin{cases} 0 & \text{if } j > n \\ LIS(i,j+1) & \text{if } A[i] \geq A[j] \\ \max\{\underbrace{LIS(i,j+1)}_{\text{NO}},\ \underbrace{1 + LIS(j,j+1)}_{\text{YES}}\} & \text{otherwise} \end{cases}$$

$A[i] < A[j]$

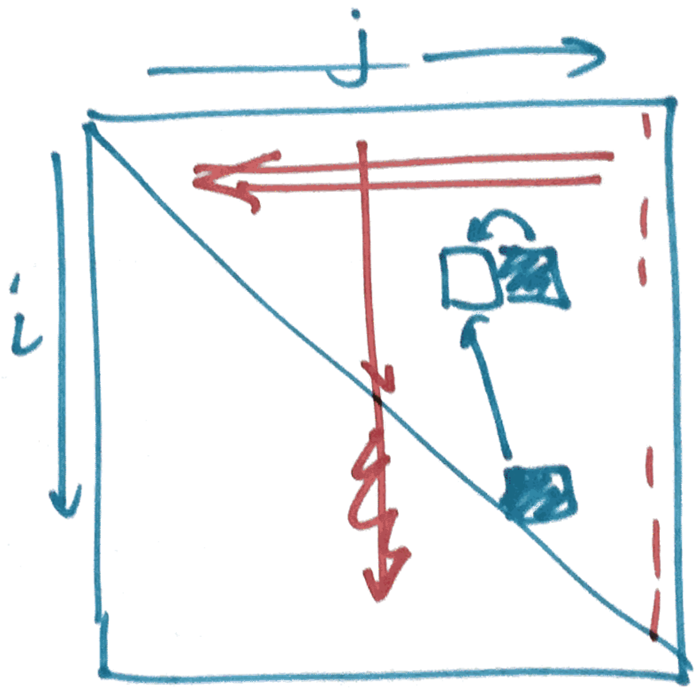
<u>LIS($A[1\,..\,n]$):</u>
$A[0] \leftarrow -\infty$ ⟨⟨*Add a sentinel*⟩⟩
for $i \leftarrow 0$ to n ⟨⟨*Base cases*⟩⟩
  $LIS[i, n+1] \leftarrow 0$

for $j \leftarrow n$ downto 1
  for $i \leftarrow 0$ to $j-1$
    if $A[i] \geq A[j]$
      $LIS[i,j] \leftarrow LIS[i,j+1]$
    else
      $LIS[i,j] \leftarrow \max\{LIS[i,j+1],\ 1 + LIS[j,j+1]\}$
return $LIS[0,1]$

$O(n^2)$



-∞
0
?
i
j



j
i

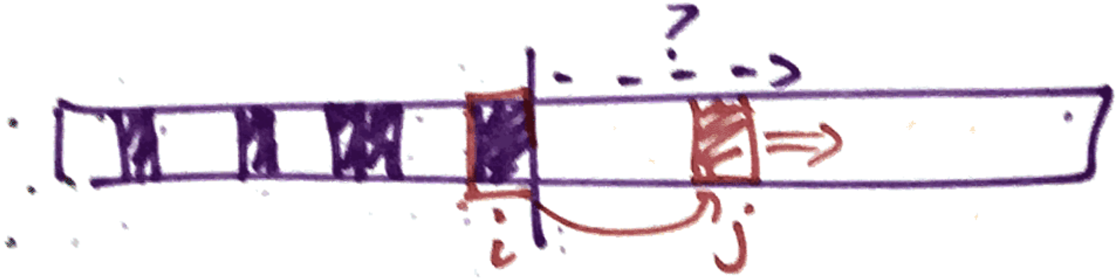
What is the next element of the longest increasing subsequence?

$$LIS2(i) := \text{length of the longest increasing subsequence of } A[i \mathinner{..} n] \text{ that starts with } A[i]$$

$$LIS2(i) = \boxed{1 + \max\left\{LIS2(j) \;\middle|\; j > i \text{ and } A[j] > A[i]\right\}} \qquad (\text{where } \max\varnothing = 0)$$

```
LIS2(A[1..n]):
  A[0] = −∞                                ⟨⟨Add a sentinel⟩⟩
  for i ← n downto 0
      LIS2[i] ← 1
      for j ← i + 1 to n
          if A[j] > A[i] and 1 + LIS2[j] > LIS2[i]
              LIS2[i] ← 1 + LIS2[j]
  return LIS2[0] − 1                       ⟨⟨Don't count the sentinel⟩⟩
```

$O(n^2)$ time


?
i
j


i

Shortest Common Supersequence

SPIDER-MAN

SUPERMAN

SUPIDER-MAN

Given $A[1..m]$ and $B[1..n]$

Find length of shortest common superseq of A and B.

Is the next char from A or B or both?

$SCS(i,j)$ = length of shortest common supersequence of $A[i..m]$ and $B[j..n]$.

$$
SCS(i,j) = \begin{cases} n-j+1 & \text{if } i>m \\ m-i+1 & \text{if } j>n \\ \min\left\{\begin{array}{l} 1+SCS(i+1,j) \\ 1+SCS(i,j+1) \\ 1+SCS(i+1,j+1) \end{array}\right\} & \text{if } A[i]=B[j] \\ \min\left\{\begin{array}{l} 1+SCS(i+1,j) \\ 1+SCS(i,j+1) \end{array}\right\} & \text{if } A[i]\neq B[j] \end{cases}
$$

$$1+SCS(i+1,j) \geq 2+SCS(i+1,j+1)$$

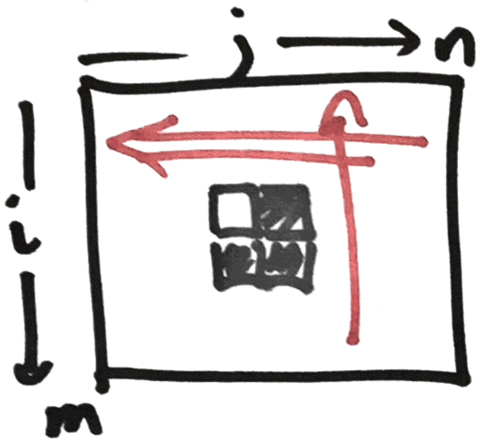
Is the next element of the shortest common supersequence an element of A or B or both?

Let $SCS(i,j) :=$ length of the shortest common supersequence of $A[i\,..\,m]$ and $B[j\,..\,n]$

$$SCS(i,j) = \begin{cases} m - j + 1 & \text{if } i > m \\ n - i + 1 & \text{if } j > n \\ 1 + SCS(i+1, j+1) & \text{if } A[i] = B[j] \\ \min \begin{Bmatrix} 1 + SCS(i+1, j) \\ 1 + SCS(i, j+1) \end{Bmatrix} & \text{if } A[i] \neq B[j] \end{cases}$$

```
SHORTESTCOMMONSUPERSEQUENCE(A[1..n], B[1..n]):
  for j ← n+1 down to 1
      SCS[m+1, j] ← n − j + 1
  for i ← n down to 1
      SCS[i, n+1] ← m − i + 1
      for j ← n down to 1
          if A[i] = B[j]
              SCS[i, j] ← 1 + SCS[i+1, j+1]
          else
              SCS[i, j] ← 1 + min{SCS[i+1, j], SCS[i, j+1]}
  return SCS(1, 1)
```

$O(n^2)$

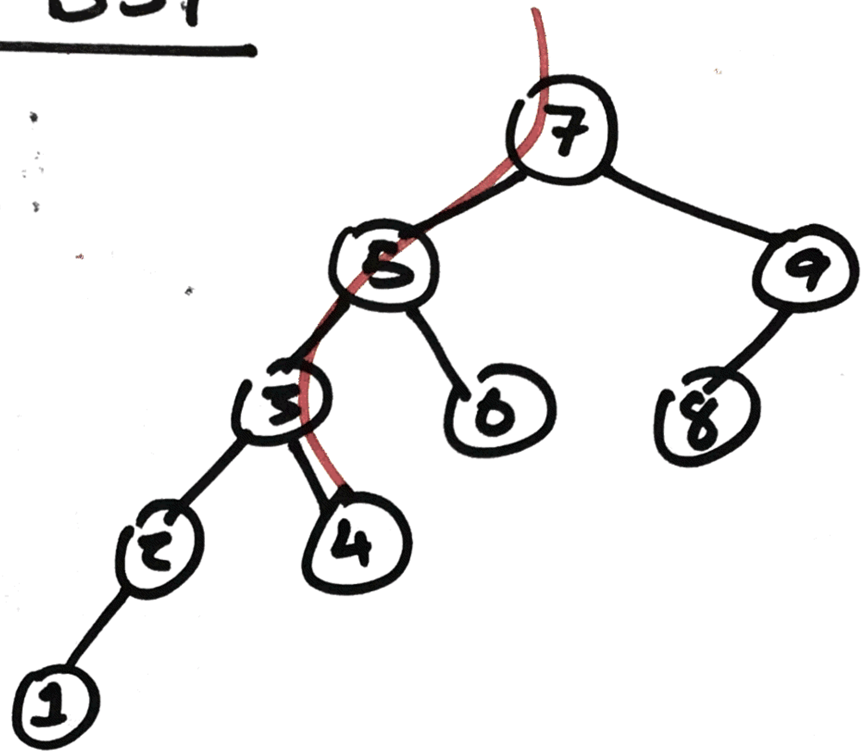

j
n
i
m

Optimal BST

Keys 1..n

Frequencies

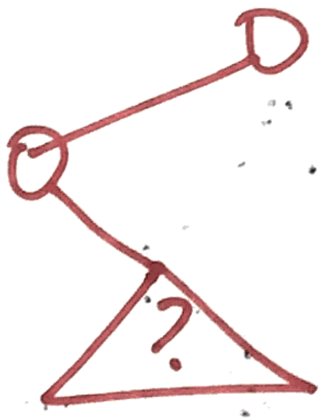
f[1..n]



$$\text{Cost}(T) = \sum_{\text{searches}} \text{depth}$$

$$= \sum_{i=1}^{n} f[i] + \underset{1..r-1}{\text{Cost}(T.\text{left})} + \underset{r+1..n}{\text{Cost}(T.\text{right})}$$

$\text{Cost}(\text{null}) = 0$



$OptCost(i,j) :=$ cost of best BST for frequencies $F[i..j]$

Which element is the root of the optimal binary tree?

$OptCost(i,k) :=$ the total cost of the optimal search tree for the frequency interval $f[i..k]$.

$$OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ \displaystyle\sum_{j=i}^{k} f[i] + \min_{i \le r \le k} \{OptCost(i, r-1) + OptCost(r+1, k)\} & \text{otherwise} \end{cases}$$

$$F(i,k) := \sum_{j=i}^{k} f[j]$$

$$F(i,k) = \begin{cases} f[i] & \text{if } i = k \\ F(i, k-1) + f[k] & \text{otherwise} \end{cases}$$

```
INITF(f[1..n]):
  for i ← 1 to n
      F[i, i−1] ← 0
      for k ← i to n
          F[i, k] ← F[i, k−1] + f[k]
```

$$
OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ F(i,k) + \min\limits_{i \le r \le k} \{OptCost(i, r-1) \; + \; OptCost(r+1, k)\} & \text{otherwise} \end{cases}
$$

```
COMPUTEOPTCOST(i, k):
  OptCost[i, k] ← ∞
  for r ← i to k
      tmp ← OptCost[i, r − 1] + OptCost[r + 1, k]
      if OptCost[i, k] > tmp
          OptCost[i, k] ← tmp
  OptCost[i, k] ← OptCost[i, k] + F[i, k]
```

```
OPTIMALBST(f[1 .. n]):
  INITF(f[1 .. n])
  for i ← 1 to n + 1
      OptCost[i, i − 1] ← 0
  for d ← 0 to n − 1
      for i ← 1 to n − d
          COMPUTEOPTCOST(i, i + d)
  return OptCost[1, n]
```